# **Final Exam (Solution)**

*This is a closed book and closed notes test.* You are not allowed to have anything on your desk other than pencil and this exam paper during the test; this includes *calculators* or *electronic assistance* of any kind – ***especially smartphones and networked devices of any kind***.

*You may not leave to go to the restroom in the first 1.5 hours.* Please go before the exam starts. After 1.5 hours, one person may go at a time. All devices must be left on your desk and you must hand your exam to the instructor while you’re away.

*You may not ask questions.* If something is confusing, write a note beside the question and explain your assumptions.

*You must show all of your work on this exam.* You will not be allowed to turn in additional sheets of paper.

*Read and sign the following statement.*  Failure to sign the statement will result in a **zero** on the exam.

*I have neither given nor received unauthorized assistance on this test. I have notified the proctor of any violations of the above policies.*

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **Problem** | **Score** |
| 1 | / 25 |
| 2 | / 25 |
| 3 | / 25 |
| 4 | / 25 |
| **Total** | **/ 100** |

*Points divided evenly among parts of a problem unless otherwise specified.*

1. (25 points) Consider the following class that implements a singly-linked list.

class List {

private:

// ListNode represents each

// node of the list

class ListNode {

public:

int item; // data in the list

ListNode \*next;

ListNode(Item a, ListNode \*n=NULL)

{

item = a;

next=n; // automatically serves as a list tail

}

};

// add head and tail pointer

ListNode \*head;

ListNode \*tail;

public:

List():head(NULL),tail(NULL) {}

// append to tail of list

void push\_back(int a);

// append to head of list

void push\_front(int a);

// remove int at tail of list

void pop\_back();

// remove int at head of list

void pop\_front();

// check if the list is empty

bool empty() const { return head==NULL; }

};

Fill in the missing parts of the following function definitions. **You may only add code to the boxes and nowhere else**, and it’s possible that some boxes should remain empty. If it should be empty, leave it empty and you will earn the designated points.

// append to tail of list

void List::push\_back(int a)

{

ListNode \*tmp = new ListNode(a);

|  |
| --- |
| (2pts)  // empty |

if (head==NULL) {

|  |
| --- |
| head = tmp;  tail = tmp;  (3pts) |

} else {

|  |
| --- |
| (3pts)  tail->next = tmp;  tail = tmp; |

}

}

// append to head of list

void List::push\_front(int a)

{

ListNode \*tmp = new ListNode(a,head);

|  |
| --- |
| (2pts)  //empty |

if (head==NULL) {

|  |
| --- |
| (3pts)  head = tail = tmp; |

} else {

|  |
| --- |
| (3pts)  head = tmp; |

}

}

// remove int at tail of list; just remove it, don’t return it

void List::pop\_back()

{

ListNode \*tmp = head, \*prev = NULL;

if (head==NULL) return;

// loop to one before the tail, then remove tail

while(tmp != tail) {

prev = tmp;

tmp = tmp->next;

}

if (prev==NULL) {

|  |
| --- |
| (2pts)  delete tmp;  head = tail = nullptr; |

} else {

|  |
| --- |
| (3pts)  delete tmp;  prev->next = nullptr;  tail = prev; |

}

}

// remove int at head of list; just remove, don’t return

void List::pop\_front()

{

ListNode \*tmp = head;

if (head==NULL) return;

if (head==tail) {

|  |
| --- |
| (2pts)  delete tmp;  head = tail = nullptr; |

} else {

|  |
| --- |
| (2pts)  head = head->next;  delete tmp; |

}

}

2. (25 points) Consider the following code and then evaluate the expressions below. If the expression has a syntax error, state that instead of evaluating the expression.

template <typename A> A add(A one, A two) { return one+two; }

class BadInt {

protected:

int x;

public:

BadInt(int a=0):x(a) {}

BadInt operator + (const BadInt &rhs) const {

return x + rhs.x;

}

BadInt operator - (const BadInt &rhs) const {

return x - rhs.x;

}

BadInt operator \* (const BadInt &rhs) const {

return x \* 2;

}

operator int () const {

return x;

}

};

BadInt a = 5, b = 6, c;

|  |  |  |  |
| --- | --- | --- | --- |
| **Points** | **Prompt** | **Expression** | **Result/Answer or Syntax error** |
| 3 | What is the value of this expression? | (int) (a-1) | Syntax error; ambiguous functions:  BadInt operator-(BadInt)  int operator-(int,int)  Accepted: 4 |
| 2 | What is the value of c.x after the expression? | c = add(a,c) | 5 |
| 2 | What is the return value of the function call? | add<int>(1.1, 2) | 3 |
| 2 | What is the value of c.x after the expression? | c = a \* b | 10 |
| 2 | What is the value of c.x after the expression? | c = a\*a - a | 5 |
| 2 | What’s the value of y? | int y = a+b; | 11 |
| 2 | What’s the value of z? | int z = a.x + b.x; | Syntax error; a.x is private. |
| 2 | What’s the value of d.x? | BadInt d; | 0 |
| 2 | What is the type created by this expression? | a + 1 | Syntax error; multiple candidate functions  BadInt operator+(BadInt)  int operator+(int,int) |
| 2 | What is the type created by this expression? | 1 + a | int |
| 2 | What functions are called in the following expression? | a + 1 | Multiple possible answers:  a.operator+(BadInt(1)) |
| 2 | What functions are called in the following expression? | 1 + a | Only possible answer:  operator+(1,a.operator int()) |

3. (25 points) Give a brief answer to each question below.

1. [4 pts] What is the big-O time complexity of the percolate-up function in a max heap?

|  |
| --- |
| log N, where N is the number of items in the heap. |

1. [3 pts] Give an example of a data structure that provides O(1) look-up.

|  |
| --- |
| Hash table |

1. [3 pts] Given an example of function overloading.

|  |
| --- |
| void foo();  void foo(int); // overloaded |

1. [3 pts] Define topological sort on a directed graph.

|  |
| --- |
| This is a sequential ordering of all the nodes in the graph, such that a source node of an edge is listed before a destination node for an edge for all edges in the graph. |

1. [3 pts] What does it mean if const is placed at the end of a member function?

|  |
| --- |
| The member function is not allowed to modify any of the class members while running, nor can it call another non-const function. |

1. [3 pts] Which keywords are used to allocate and deallocate heap memory?

|  |
| --- |
| new, delete |

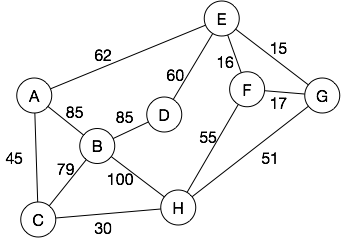
1. [3 pts] What are a class constructor and destructor?

|  |
| --- |
| Constructor: runs when an object is allocated to initialize fields;  Desctructor: runs with an object is deallocated to free objects it allocated; |

1. [3 pts] Name a class in the standard template library that implements a balanced binary search tree.

|  |
| --- |
| Any of these: map, set |

4. (25 points) Consider the following graph:



(a) [5 points] Give a depth-first search pre-order for the graph above starting at node A.

|  |
| --- |
| Any legal DFS order is fine:  A E F G H C B D |

(b) [5 points] Give a breadth-first search order for the graph above starting at node A.

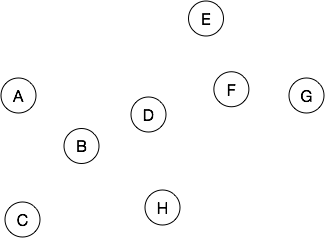
|  |
| --- |
| Any legal BFS order is fine:  A E B C D F G H |

(c) [10 points] Calculate the Minimum Spanning Tree for the graph above. Show your work in the table below, and add the edges of the Minimum Spanning Tree to the graph on the following page.

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge** | **Weight** | **Already connected?** | **Vertex Sets** |
| E-G | 15 | No | {E,G} |
| E-F | 16 | No | {E,F,G} |
| F-G | 17 | Yes |  |
| C-H | 30 | No | {C,H}, {E,F,G} |
| A-C | 45 | No | {A,C,H}, {E,F,G} |
| H-G | 51 | No | {A,C,E,F,G,H} |
| H-F | 55 | Yes |  |
| D-E | 60 | No | {A,C,D,E,F,G,H} |
| A-E | 62 | Yes |  |
| B-C | 79 | No | {A,B,C,D,E,F,G,H} |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(d) [5 points] Add the edges of the Minimum Spanning Tree to the graph below.

(Draw in MST edges - not shown)



**C++ Keywords**

In common with C:

auto const double float int short struct unsigned  
break continue else for long signed switch void  
case default enum goto register sizeof typedef volatile  
char do extern if return static union while

Unique to C++:

asm dynamic\_cast namespace reinterpret\_cast try  
bool explicit new static\_cast typeid  
catch false operator template typename  
class friend private this using  
const\_cast inline public throw virtual  
delete mutable protected true wchar\_t

Reserved words:

and bitand compl not\_eq or\_eq xor\_eq  
and\_eq bitor not or xor

**ASCII Table**

